# Computer Vision HW 10: Report

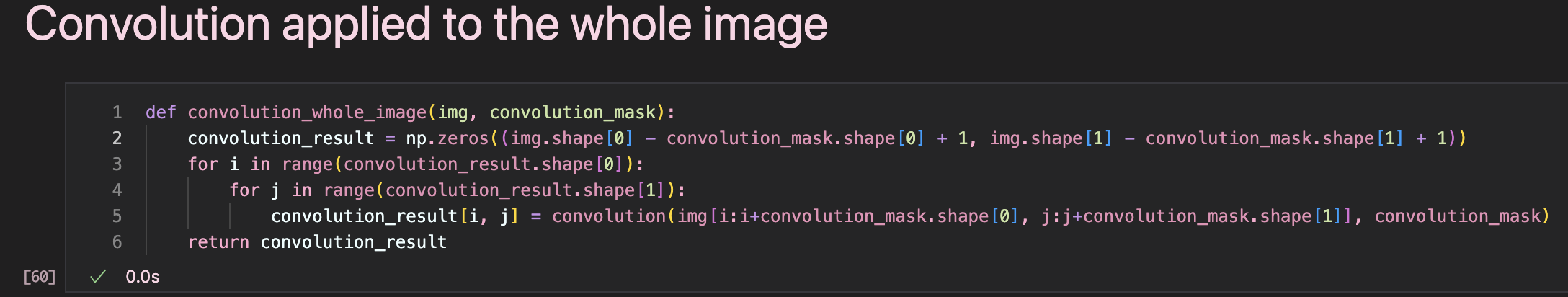
[[1]](#footnote-1) [[2]](#footnote-2)[[3]](#footnote-3) [[4]](#footnote-4)[[5]](#footnote-5)

## Code Structure

1. Convolution
   * Convolution(): convolve a particular neighborhood
   * Convolution\_whole\_image(): iteratively apply the convolution function to each neighborhood within the image
2. Convolution result Laplacian output
   * Convolution\_result\_to\_Laplacian\_output()
3. Laplacian result zero crossing
   * Laplacian\_output\_to\_result\_image
4. Function that each corresponds (a) ~ (e)

## Code details

For the first part, first we have the same function as used in the previous homework, but a modification is added, which is adding another Convolution\_whole\_image() function, so that we can call this function and get the convolution result directly, without explicitly looping though each neighborhood in the functions (a) ~ (e).



In this function, we initialize a zero-filled array, with size that ensures the last row / column of the kernel can be placed within the original image. For example, if we have a 2x2 kernel, we cannot put the top row on the last row of our image.

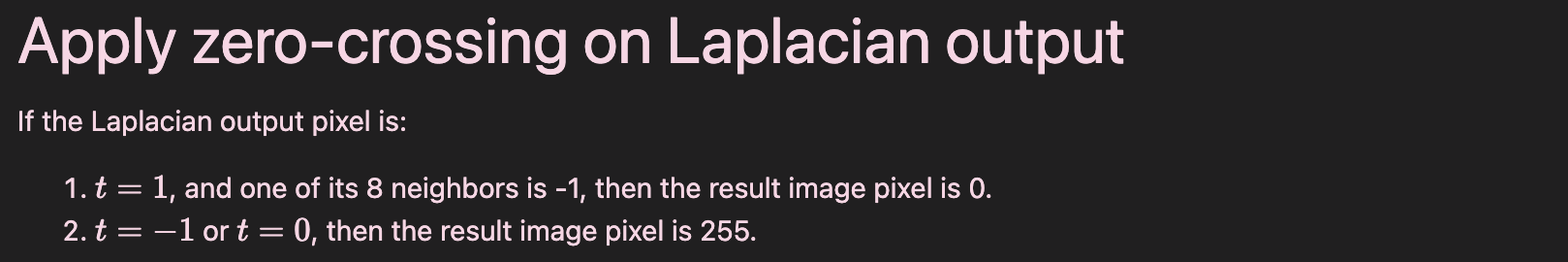
Next, the convolution result of each pixel is calculated by calling the convolution function.

For the second part, we need to transform this result (the whole image after doing convolution) into Laplacian, which means the function below will output an array full of values of :

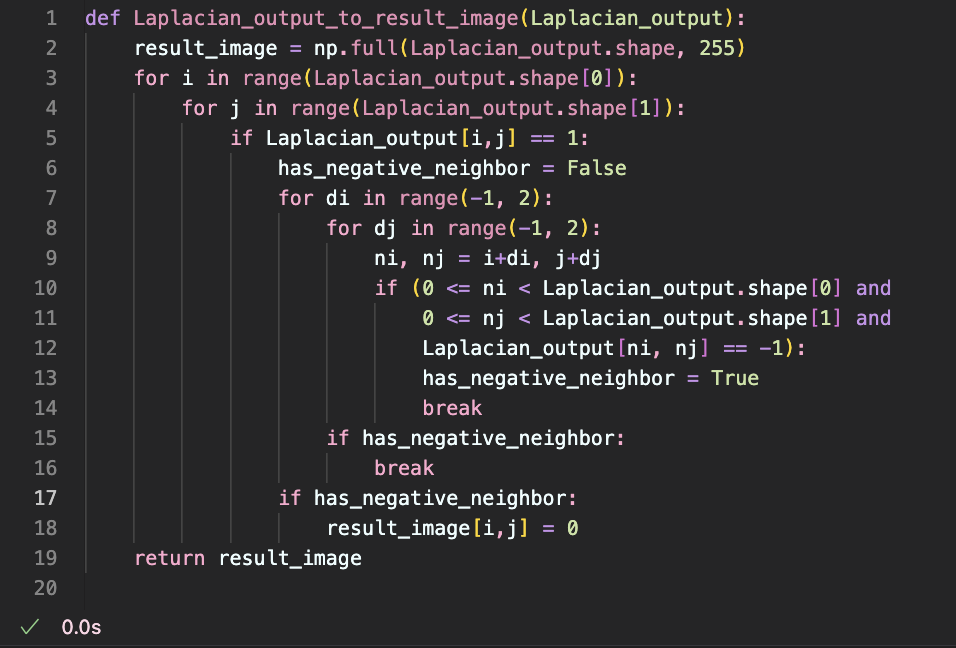


This function is quite simple, we just follow the definition on the ppt, to set the values of to .

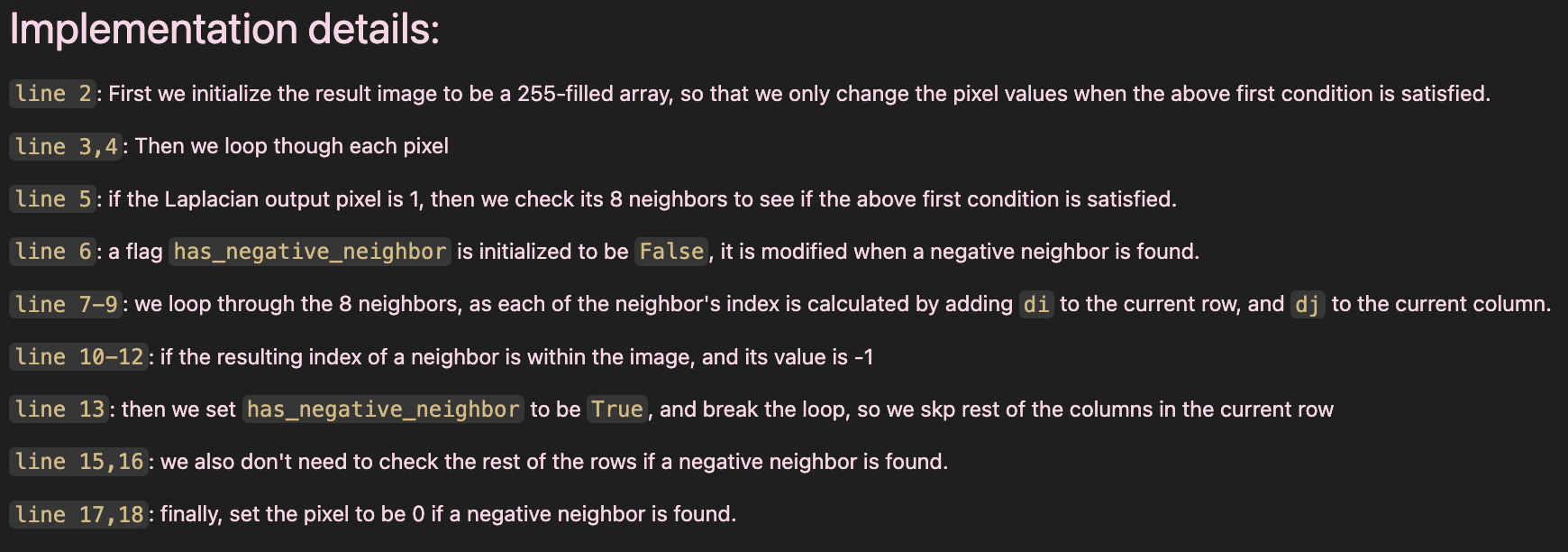
The third part is to do zero-crossing, we also follow the definition on the ppt to transform values of into intensity values :



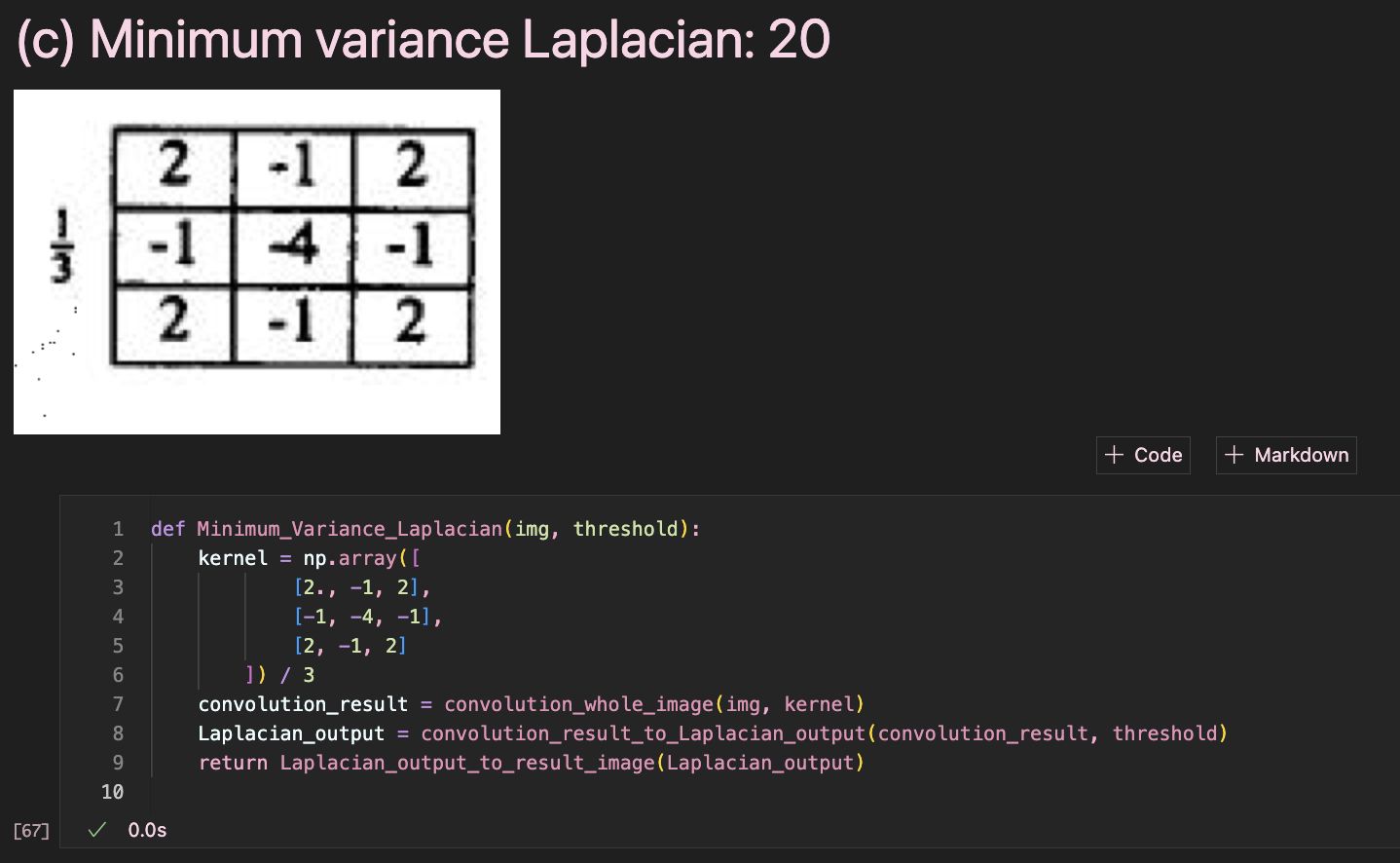
The code is as follows:



A really detailed explanation for each line is in the associated markdown cell in the source code file:

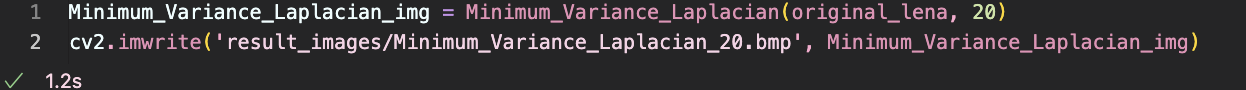


For the last part, which is implementing the functions corresponding to functions (a) ~ (e), they all have similar structure, take (c) for example:



The image in the markdown cell is the kernel we should use as in ppt, in the function, we first define this kernel, then what we need to do is to follow the steps of our previous 3 parts:

Get convolution result get laplacian result zero crossing result (the resulting image shown in the report)



Finally, call the function with required threshold, and save the result by cv2.imwrite().

1. (a) Laplacian mask 1, threshold = 15 [↑](#footnote-ref-1)
2. (b) Laplacian mask 2, threshold = 15 [↑](#footnote-ref-2)
3. (c) Minimum variance Laplacian, threshold = 20 [↑](#footnote-ref-3)
4. (d) Laplace of Gaussian, threshold = 3000 [↑](#footnote-ref-4)
5. (e) Difference of Gaussian, threshold = 1 [↑](#footnote-ref-5)